REMARKS

Claims 37-44 stand rejected as being anticipated by *Montrym* (US Application No. 2003/0103054). Applicants respectfully request reconsideration of the rejections against claims 37-44 based on the following remarks. In particular, Applicants' request that the terms "*conventional vertex processing*" and *"application-programmable vertex processing system"* be considered in light of the specification.

New claim 45 has been added.

Definition of "conventional vertex processing"

In the office action dated May 5, 2004, the Examiner requested clarification on the meaning of "conventional vertex processing." This term, or slight variations of the term, is used in each of the independent claims presented for examination. This term is also used throughout the specification in a specific way. For example, pages 1 and 2 of the specification describe what is meant by the term "conventional vertex processing:"

> Conventional vertex processing for three-dimensional (3-D) graphics programming application program interfaces (APIs) such as Open Graphics Library (OpenGL®) and D3D™ provide support for per-vertex lighting, position transformation, and texture coordinate generation. The computations provided by such conventional vertex processing are routinely implemented by 3-D graphics hardware that greatly accelerates these operations.
>
> One drawback of the aforementioned conventional vertex processing is that it is configurable, but not programmable. When using conventional vertex processing, an application can enable and disable various options, set transformation matrices, lighting, and texture coordinate generation parameters. However, such applications are limited to the set of computations provided by the conventional vertex processing feature set.
>
> While the feature set has been gradually extended over time to support multiple texture units, and more texture coordinate generation modes and vertex blending schemes, the conventional

vertex processing model is still fundamentally configurable, not programmable.

Conventional vertex processing assigns names to per-vertex quantities such as "position," "color," and "surface normal." These names convey a sense of how the quantities are processed by conventional vertex processing. For example, surface normal are used for lighting vertices. The quantities' meaning is directly tied to the operations performed with the quantity by conventional vertex processing. Similarly, other quantities such as "light position," "light color," and "model view matrix" are named to convey how these quantities are used by conventional vertex processing.

Existing applications use API commands named based on the conventions of conventional vertex processing. For example, a vertex may be set in the manner show in Table 1.

Table 1

```
glNormal3f (xnor, ynor, znor);
glColor3f (red, green, blue);
glVertex3F (xpos, ypos, zpos);
```

Definition of "application-programmable vertex processing system"

In the office action dated May 5, 2004, the Examiner requested clarification on the meaning of "application-programmable vertex processing." This term, or slight variations of the term, is used in each of the independent claims presented for examination. This term is also used throughout the specification in a specific way. For example, pages 2 and 3 of the specification describe what is meant by the term "application-programmable vertex processing system:"

In contrast with conventional vertex processing, application-programmable vertex processing has no pre-existing meaning for the quantities used to process vertices. Instead, there is simply a predetermined amount of numbered per-vertex quantities (per-vertex variables) and a predetermined amount of state numbered

quantities (per-vertex constants). How these quantities are used to process the vertices depends on the application-supplied vertex program's instruction sequence.

For example, a vertex would be set in the manner set forth in Table 1A.

Table 1A

```
glVertexAttrib3fNV(2, xnor, ynor, znor);
glVertexAttrib3fNV(3, red, green, blue);
glVertexAttrib3fNV(0, xpos, ypos, zpos);
```

Prior art techniques for extending conventional vertex processing generally require adding more modes, state, and per-vertex attributes. This lead to per-vertex attributes beyond the standard OpenGL per-vertex attributes (position, normal, color, texture coordinates, etc.) Examples of the new (extended) attributes are secondary color, fog coordinate, weights (for vertex blending), and additional texture coordinate sets.

While application-programmable vertex processing provides tremendous flexibility in comparison to conventional vertex processing, 3-D applications must, however, assign their own meaning to vertex processing quantities rather than have meanings assigned by the conventions of conventional vertex processing. Because vertex programs assign the "meaning" to vertex attributes based on how the program uses the various vertex attributes, it makes little sense to give the vertex attributes conventional names. Application-programmable vertex processing considers vertex attributes "generic" numbered quantities.

CONCLUSION

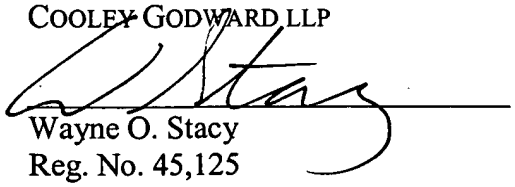
In view of the foregoing, Applicants respectfully request consideration of the new claims. The Examiner is requested to call Applicants' representative if any question or comments arise.

The Commissioner is hereby authorized to charge any appropriate fees under 37 C.F.R. §§1.16, 1.17, and 1.21 that may be required by this paper, and to credit any overpayment, to Deposit Account No. 03-3117.

COOLEY GODWARD LLP
Attention: Patent Group
Five Palo Alto Square
3000 El Camino Real
Palo Alto, CA 94306-2155
Tel: (720) 566-4125
Fax: (720) 566-4099

Respectfully submitted,

COOLEY GODWARD LLP

By: 
Wayne O. Stacy
Reg. No. 45,125